REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 09-03-2012	2. REPORT TYPE Related Material	3. DATES COVERED (From - To) -

4. TITLE AND SUBTITLE Finding Top-k "Unexplained" Activities in Video	5a. CONTRACT NUMBER W911NF-09-1-0206
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 611102
6. AUTHORS Massimiliano Albanese, Cristian Molinaro, Fabio Persia, Antonio Picariello, V.S. Subrahmanian	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES University of Maryland - College Park, Research Administration, 3112 Lee Building, College Park, MD 20742 -5141	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 56057-NS.28

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
Most past work on identifying unexpected activities in video has focused on looking for specific patterns of
anomalous
activities. In this paper, we consider the situation where we have a known set A of activities (normal and abnormal)
that we wish
to monitor. However, in addition, we wish to identify abnormal activities that have not been previously considered

15. SUBJECT TERMS
unexplained activity, stochastic automata, algorithms, modeling, probability, temporal constraints,

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			V.S. Subrahmanian
UU	UU	UU	UU		19b. TELEPHONE NUMBER 301-405-6724

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

Finding Top-k "Unexplained" Activities in Video

ABSTRACT

Most past work on identifying unexpected activities in video has focused on looking for specific patterns of anomalous
activities. In this paper, we consider the situation where we have a known set A of activities (normal and abnormal) that we wish
to monitor. However, in addition, we wish to identify abnormal activities that have not been previously considered or encountered,
i.e. they are not in A. We formally define the probability that a video sequence is unexplained (totally or partially) w.r.t. A. We
develop efficient algorithms to identify the top-k Totally and Partially Unexplained Activities in a video w.r.t. A. Our algorithms use
neat mathematical properties of the definitions for efficiency. We describe experiments using two real-world datasets showing
that our approach works well in practice in terms of both running time and accuracy.

Finding Top-k “Unexplained” Activities in Video

Massimiliano Albanese, Cristian Molinaro, Fabio Persia, Antonio Picariello, V. S. Subrahmanian

Abstract—Most past work on identifying unexpected activities in video has focused on looking for specific patterns of anomalous activities. In this paper, we consider the situation where we have a known set $\mathcal{A}$ of activities (normal and abnormal) that we wish to monitor. However, in addition, we wish to identify abnormal activities that have not been previously considered or encountered, i.e. they are not in $\mathcal{A}$. We formally define the probability that a video sequence is unexplained (*totally* or *partially*) w.r.t. $\mathcal{A}$. We develop efficient algorithms to identify the top-k *Totally* and *Partially Unexplained Activities* in a video w.r.t. $\mathcal{A}$. Our algorithms use neat mathematical properties of the definitions for efficiency. We describe experiments using two real-world datasets showing that our approach works well in practice in terms of both running time and accuracy.

1 Introduction

Video surveillance is omnipresent. For instance, airport baggage areas are continuously monitored for suspicious activities. In crime-ridden neighborhoods, police often monitor streets and parking lots using video surveillance. In Israel, highways are monitored by a central authority for suspicious activities. However, all these applications search for *known* activities – activities that have been identified in advance as being either “normal” or “abnormal”. For instance, in the highway application, security officers may look both for normal behavior (e.g. driving along the highway in a certain speed range unless traffic is slow) as well as “suspicious” behavior (e.g. stopping the car near a bridge, taking a package out and leaving it on the side of the road before driving away).

In this paper, we are given a set $\mathcal{A}$ of activity definitions expressed as stochastic automata with temporal constraints (extending [1]).[1] $\mathcal{A}$ can contain “normal” activities or “suspicious” activities or both. We then try to find video sequences that are not “sufficiently explained” by any of the activities in $\mathcal{A}$. For instance, in an airport, we may know of certain patterns that are suspicious, but we may also know that there are many activity patterns that a criminal/terrorist may use that we cannot possibly predict. Such “unknown” activities are “unexplained” in our framework.

We achieve this via a possible-worlds based model and define the probability that a sequence of video is *totally* (or *partially*) unexplained. Based on this, users can specify a probability threshold and look for all sequences that are totally (or partially) unexplained with a probability exceeding the threshold. We then show different important properties we can leverage to make the search of unexplained activities more efficient. We define algorithms to find top-k totally and partially unexplained activities. We develop a prototype implementation and report on experiments using two data sets showing that the algorithms work well in practice, both from an efficiency perspective and an accuracy perspective.

The paper starts (Section 2) with an overview of related work. Section 3 provides basic definitions of stochastic activities slightly extending [1]. Section 4 defines the probability that a video sequence is totally (or partially) unexplained. We also define the problem of finding the top-k (totally or partially) unexplained activities and classes. Section 5 derives theorems that enable fast search for totally and partially unexplained video sequences. Section 6 presents algorithms for solving the problems introduced in Section 4. Section 7 describes our experiments. The paper concludes in Section 8. [2]

2 Related Work

A Priori Definitions. Several researchers have studied how to search for specifically defined patterns of normal/abnormal activities [2]. [3] studies how HMMs can be used to recognize complex activites, while [4] and [5] use coupled HMMs. [6] uses Dynamic Bayesian Networks (DBNs) to capture causal relationships between observations and hidden states. [1] developed a stochastic automaton based language to detect activities in video, while [7] presented an HMM-based algorithm. *In contrast, this paper starts with a set $\mathcal{A}$ of activity models (corresponding to normal/abnormal activities) and finds video sequences that are not sufficiently explained by the models in $\mathcal{A}$.*

- *M. Albanese is with the Department of Applied Information Technology, George Mason University, Nguyen Engineering Building, Fairfax, VA 22030. E-mail: malbanes@gmu.edu*
- *C. Molinaro and V. S. Subrahmanian are with the Department of Computer Science and UMIACS, University of Maryland, A.V. Williams Building, College Park, MD 20742. E-mail: {molinaro, vs}@umiacs.umd.edu*
- *F. Persia and A. Picariello, are with Dipartimento di Informatica e Sistemistica, Università di Napoli “Federico II”, Via Claudio 21, 80125 Napoli, Italy. E-mail: {fabio.persia, picus}@unina.it*

1. Stochastic automata are closely related to both HMM and DBN framework that have been studied extensively in activity detection.

2. All proofs are reported in a detachable appendix included for the convenience of the reviewers.

Such unexplained sequences reflect activity occurrences for which no model exists a priori.

Learning and then detecting abnormality. Several researchers first learn normal activity models and then detect abnormal/unusual events. [8] suggests a semi-supervised approach to detect abnormal events that are rare, unexpected, and relevant. We do not require "unexplained" events to either be rare or relevant. [9] uses HMMs to detect rare events, while [10] defines an anomaly as an atypical behavior pattern that is not represented by sufficient samples in a training dataset and satisfies an abnormal pattern. [11] defines abnormality as unseen or rarely occurring events — an initial video is used to learn normal behaviors. [12] shows how to detect users with abnormal activities from sensors attached to human bodies. An abnormal activity is defined as "an event that occurs rarely and has not been expected in advance". Abnormal activities become normal when they start occur more often. The same notion of abnormal activity is considered in [13] and [14]. [15] learns patterns of activities overtime in an unsupervised way. [16] deals with detecting individual anomalies in crowd scenes — an anomaly is defined as a rare or infrequent behavior compared to all other behaviors. The normality/abnormality of an individual behavior is evaluated w.r.t. a specific context. Then, usual activities are accepted as normal and deviant activity patterns are flagged as abnormal. All these approaches first learn normal activity models and then detect abnormal/unusual events. These papers differ from ours because they consider rare events to be abnormal. In contrast, we consider activities to be abnormal even if they are not rare. For example, if a new way to break into cars has proliferated during the past month, then we want to flag those activities as "unexplained" even if they are no longer rare. In addition, if a model exists for a rare activity, we would flag it as normal, while many of these frameworks would not.

Similarity-based abnormality. [17] proposes an unsupervised technique in which no activity model is required a priori and no explicit models of normal activities are built. Each event in the video is compared with all other observed events to determine how many similar events exist. Unusual events are events for which there are no similar events in the video. Hence, this work also considers unusual activity as a rare event and a large number of observations is required to verify if an activity is indeed unusual. [18] uses a similar approach: a scene is considered anomalous when the maximum similarity between the scene and all previously viewed scenes is below a threshold. This is also similar to [19] where frequently occurred patterns are normal and patterns that are dissimilar from most patterns are anomalous. [20] learns trajectory prototypes and detects anomalous behaviors when visual trajectories deviate from the self-learned representations of typical behaviors. In [3], activities performed by a group of moving and interacting objects are modeled as shapes and abnormal activities are then defined as a change in the shape activity model.

Other relevant work. [21] develops an algorithm that collects low-level scene observations representing routine activities. Unusual events are detected by monitoring the scene with *monitors* which extracts local low-level observations from the video stream. Given a new observation, the monitor computes the likelihood of this observation with respect to the probability distribution of prior observations. If the likelihood falls below a certain threshold, then the monitor outputs an alert. The local alerts issued by the monitors are then combined. [22] automatically learns high frequency events (taking spatio-temporal aspects into account) and declares them normal; then, events deviating from these rules are anomalies.

3 Basic Activity Model

This section extends the stochastic activity model of [1] (though we make no claims of novelty for this). We assume the existence of a finite set $\mathcal{S}$ of *action symbols*, corresponding to atomic actions that can be detected by image understanding methods.

Definition 3.1 (Stochastic activity): A *stochastic activity* is a labeled directed graph $A = (V, E, \delta, \rho)$ where

- V is a finite set of nodes labeled with action symbols from $\mathcal{S}$;
- $E \subseteq V \times V$ is a set of edges;
- $\delta : E \to \mathbb{N}^+$ associates, with each edge $\langle v_i, v_j \rangle$, an upper bound on the time that can elapse between v_i and v_j;
- ρ is a function that associates, with each node $v \in V$ having out-degree 1 or more, a probability distribution on $\{\langle v, v' \rangle \mid \langle v, v' \rangle \in E\}$, i.e., $\sum_{\langle v,v' \rangle \in E} \rho(\langle v, v' \rangle) = 1$;
- $\{v \in V \mid \nexists\, v' \in V \;s.t.\; \langle v', v \rangle \in E\} \neq \emptyset$, i.e., there exists at least one *start node* in the activity definition;
- $\{v \in V \mid \nexists\, v' \in V \;s.t.\; \langle v, v' \rangle \in E\} \neq \emptyset$, i.e., there exists at least one *end node* in the activity definition.

Figure 1 shows an example of stochastic activity modeling deposits at an Automatic Teller Machine (ATM). Each edge e is labeled with $(\delta(e), \rho(e))$. For instance, the two edges starting at node insertCard mean that there is a 50% probability of going to node insertChecks and a 50% probability of going to node insertCash from node insertCard. In addition, it is required that insertChecks and insertCash follow insertCard within 2 and 1 time units, respectively. For the purpose of this paper, this example is simplified (e.g., we avoided talking about the customer typing on the keypad, etc.). In general, actions can be easily detected by either an image processing algorithm (e.g. detectPerson would check if a person is present in the image) or a sensor (e.g. to detect if insertCard holds).

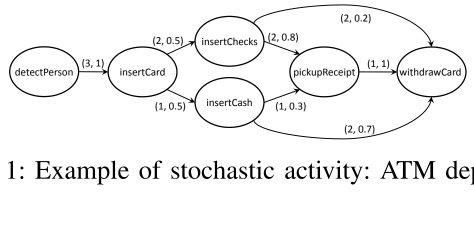


Fig. 1: Example of stochastic activity: ATM deposit

This framework extends [1] by adding the function δ which expresses a constraint on the maximum "temporal distance" between two actions in an activity.

Definition 3.2 (Stochastic activity instance): An *instance* of a stochastic activity (V, E, δ, ρ) is a sequence $\langle s_1, \ldots, s_m \rangle$ of nodes in V such that

- $\langle s_i, s_{i+1} \rangle \in E$ for $1 \leq i < m$;
- $\{s \mid \langle s, s_1 \rangle \in E\} = \emptyset$, i.e., s_1 is a start node; and
- $\{s \mid \langle s_m, s \rangle \in E\} = \emptyset$, i.e., s_m is an end node.

The probability of the instance is $\prod_{i=1}^{m-1} \rho(\langle s_i, s_{i+1} \rangle)$.

Thus, an instance of a stochastic activity A is a path in A from a start node to an end node. In Figure 1, $\langle$detectPerson, insertCard, insertCash, withdrawCard$\rangle$ is an instance with probability 0.35. Throughout this paper, we assume an arbitrary but fixed set $\mathcal{A}$ of stochastic activities.

A *video* is a finite sequence of frames. Each frame f has an associated timestamp, denoted $f.ts$; without loss of generality, we assume timestamps to be positive integers. A *labeling* ℓ of a video v is a mapping $\ell : v \to 2^{\mathcal{S}}$ that takes a video frame $f \in v$ as input, and returns a set of action symbols $\ell(f) \subseteq \mathcal{S}$ as output. Intuitively, a labeling can be computed via an appropriate suite of image processing algorithms and specifies the actions detected in each frame of a video.

Example 3.1: Consider a video $v = \langle f_1, f_2, f_3, f_4, f_5 \rangle$, with $f_i.ts = i$ for $1 \leq i \leq 5$. A possible labeling ℓ of v is: $\ell(f_1) = \{\text{detectPerson}\}$, $\ell(f_2) = \{\text{insertCard}\}$, $\ell(f_3) = \{\text{insertCash}\}$, $\ell(f_4) = \{\text{withdrawCash}\}$, $\ell(f_5) = \{\text{withdrawCard}\}$.

Suppose $S_1 = \langle a_1, \ldots, a_n \rangle$ and $S_2 = \langle b_1, \ldots, b_m \rangle$ are two sequences. S_2 is a *subsequence* of S_1 iff there exist $1 \leq j_1 < j_2 < \ldots < j_m \leq n$ s.t. $b_i = a_{j_i}$ for $1 \leq i \leq m$. If $j_i = j_{i+1} - 1$ for $1 \leq i < m$, then S_2 is a *contiguous* subsequence of S_1. We write $S_1 \cap S_2 \neq \emptyset$ iff S_1 and S_2 have a common element and write $e \in S_1$ iff e is an element appearing in S_1. The *concatenation* of S_1 and S_2, i.e., the sequence $\langle a_1, \ldots, a_n, b_1, \ldots, b_m \rangle$, is denoted by $S_1 \cdot S_2$. Finally, we use $|S_1|$ to denote the length of S_1, that is, the number of elements in S_1.

We now define an occurrence of a stochastic activity in a video.

Definition 3.3 (Activity occurrence): Let v be a video, ℓ a labeling of v, and $A = (V, E, \delta, \rho)$ a stochastic activity. An *occurrence* o of A in v w.r.t. ℓ is a sequence $\langle (f_1, s_1), \ldots, (f_m, s_m) \rangle$ such that

- $\langle f_1, \ldots, f_m \rangle$ is a subsequence of v,
- $\langle s_1, \ldots, s_m \rangle$ is an instance of A,
- $s_i \in \ell(f_i)$, for $1 \leq i \leq m$, and [3]
- $f_{i+1}.ts - f_i.ts \leq \delta(\langle s_i, s_{i+1} \rangle)$, for $1 \leq i < m$.

The probability of o, denoted $p(o)$, is the probability of the instance $\langle s_1, \ldots, s_m \rangle$.

When concurrently monitoring multiple activities, shorter activity instances generally tend to have higher probability. To remedy this, we normalize occurrence probabilities by introducing the relative probability $p^*(o)$ of an occurrence o of activity A as $p^*(o) = \frac{p(o)}{p_{max}}$, where p_{max} is the highest probability of any instance of A.

Example 3.2: Consider the video and the labeling of Example 3.1. An occurrence of the activity of Figure 1 is $o = \langle (f_1, \text{detectPerson}), (f_2, \text{insertCard}), (f_3, \text{insertCash}), (f_5, \text{withdrawCard}) \rangle$, and $p^*(o) = 0.875$.

We use $\mathcal{O}(v, \ell)$ to denote the set of all activity occurrences in v w.r.t. ℓ. Whenever v and ℓ are clear from the context, we write $\mathcal{O}$ instead of $\mathcal{O}(v, \ell)$.

4 Unexplained Activity Probability Model

This section defines the probability that a video sequence is unexplained by $\mathcal{A}$. We note that the occurrence of an activity in a video can involve conflicts. For instance, consider the activity occurrence o in Example 3.2 and suppose there is a second activity occurrence o' such that $(f_1, \text{detectPerson}) \in o'$. In this case, there is an implicit conflict because $(f_1, \text{detectPerson})$ belongs to both occurrences, but in fact, detectPerson can only belong to one activity occurrence, i.e. though o and o' may both have a non-zero probability, the probability that these two activity occurrences coexist is 0. Formally, we say two activity occurrences o, o' *conflict*, denoted $o \nsim o'$, iff $o \cap o' \neq \emptyset$. We now use this to define possible worlds.

Definition 4.1 (Possible world): A *possible world* for a video v and a labeling ℓ is a subset w of $\mathcal{O}$ s.t. $\nexists o_i, o_j \in w, o_i \nsim o_j$.

Thus, a possible world is a set of activity occurrences which do not conflict with one another, i.e., an action symbol in a frame cannot belong to two distinct activity occurrences in the same world. We use $\mathcal{W}(v, \ell)$ to denote the set of all possible worlds for a video v and a labeling ℓ; whenever v and ℓ are clear from the context, we simply write $\mathcal{W}$.

Example 4.1: Consider a video with two conflicting occurrences o_1, o_2. There are 3 possible worlds: $w_0 = \emptyset$, $w_1 = \{o_1\}$, and $w_2 = \{o_2\}$. Note that $\{o_1, o_2\}$ is not a world as $o_1 \nsim o_2$. Each world represents a way of explaining what is observed. The first world corresponds to the case where nothing is explained, the second and third worlds correspond to the scenarios where we use one of the two possible occurrences to explain the observed action symbols.

Note that any subset of $\mathcal{O}$ not containing conflicting occurrences is a legitimate possible world — possible worlds are not required to be maximal w.r.t. $\subseteq$. In the above example, the empty set is a possible world even though there are two other possible worlds $w_1 = \{o_1\}$ and $w_2 = \{o_2\}$ which are supersets of it. The reason is that o_1 and o_2 are uncertain, so the scenario where neither o_1 nor o_2 occurs is a legitimate one. We further illustrate this point below.

3. With a slight abuse of notation, we use s_i to refer to both node s_i and the action symbol labeling it.

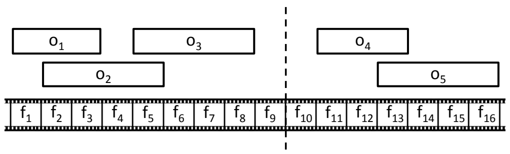


Fig. 2: Conflict-Based Partitioning of a video

Example 4.2: Suppose we have a video where a single occurrence o has $p^*(o) = 0.6$. In this case, it is natural to say that there are two possible worlds $w_0 = \emptyset$ and $w_1 = \{o\}$ and expect the probabilities of w_0 and w_1 to be 0.4 and 0.6, respectively. By restricting ourselves to maximal possible worlds only, we would have only one possible world, w_1, whose probability is 1, which is wrong.

We use $\overset{*}{\nsim}$ to denote the transitive closure of $\nsim$. Clearly, $\overset{*}{\nsim}$ is an equivalence relation and determines a partition of $\mathcal{O}$ into equivalence classes $\mathcal{O}_1, \ldots, \mathcal{O}_m$.

Example 4.3: Suppose we have a video $v = \langle f_1, \ldots, f_{16} \rangle$ and a labeling ℓ such that five occurrences o_1, o_2, o_3, o_4, o_5 are detected as depicted in Figure 2, that is, $o_1 \nsim o_2$, $o_2 \nsim o_3$, and $o_4 \nsim o_5$. There are two equivalence classes determined by $\overset{*}{\nsim}$, namely $\mathcal{O}_1 = \{o_1, o_2, o_3\}$ and $\mathcal{O}_2 = \{o_4, o_5\}$.

The equivalence classes determined by $\overset{*}{\nsim}$ lead to a conflict-based partitioning of a video.

Definition 4.2 (Conflict-Based Partitioning): Let v be a video, ℓ a labeling, and $\mathcal{O}_1, \ldots, \mathcal{O}_m$ the equivalence classes determined by $\overset{*}{\nsim}$. A *Conflict-Based Partitioning* (CBP) of v (w.r.t. ℓ) is a sequence $\langle (v_1, \ell_1), \ldots, (v_m, \ell_m) \rangle$ such that:

- $v_1 \cdot \ldots \cdot v_m = v$;
- ℓ_i is the restriction of ℓ to v_i, i.e., it is a labeling of v_i s.t. $\forall f \in v_i, \ell_i(f) = \ell(f)$, for $1 \leq i \leq m$; and
- $\mathcal{O}(v_i, \ell_i) = \mathcal{O}_i$, for $1 \leq i \leq m$.

The v_i's are called *segments*.

Example 4.4: A CBP of the video in Example 4.3 is $\langle (v_1, \ell_1), (v_2, \ell_2) \rangle$, where $v_1 = \langle f_1, \ldots, f_9 \rangle$, $v_2 = \langle f_{10}, \ldots, f_{16} \rangle$, ℓ_1 and ℓ_2 are the restrictions of ℓ to v_1 and v_2, respectively. Another partitioning of the same video is the one where $v_1 = \langle f_1, \ldots, f_{10} \rangle$ and $v_2 = \langle f_{11}, \ldots, f_{16} \rangle$.

Thus, activity occurrences determine a set of possible worlds (intuitively, different ways of explaining the video). We wish to find a probability distribution over all possible worlds that (i) is consistent with the relative probabilities of the occurrences, and (ii) takes conflicts into account. We assume the user specifies a function $Weight : \mathcal{A} \rightarrow \mathbb{R}^+$ which assigns a weight to each activity and prioritizes the importance of the activity.[4] The weight of an occurrence o of activity A is the weight of A. We use $C(o)$ to denote the set of occurrences conflicting with o, i.e., $C(o) = \{o' \mid o' \in \mathcal{O} \land o' \nsim o\}$. Note that $o \in C(o)$; and $C(o) = \{o\}$ when o does not conflict with any other occurrence. Finally, we assume that activity occurrences belonging to different segments are independent events. Suppose p_i denotes the (unknown) probability of world w_i. As we know the probability of occurrences, and as each occurrence occurs in certain worlds, we can induce a set of nonlinear constraints that will subsequently be used to learn the values of the p_i's.

4. For instance, highly threatening activities may be assigned a high weight.

Definition 4.3: Let v be a video, ℓ a labeling, and $\mathcal{O}_1, \ldots, \mathcal{O}_m$ the equivalence classes determined by $\overset{*}{\nsim}$. We define the non-linear constraints $NLC(v, \ell)$ as follows:

$$\begin{cases} p_i \geq 0, \quad \forall w_i \in \mathcal{W} \\ \sum_{w_i \in \mathcal{W}} p_i = 1 \\ \sum_{w_i \in \mathcal{W} \, s.t. \, o \in w_i} p_i = p^*(o) \cdot \frac{Weight(o)}{\sum_{o_j \in C(o)} Weight(o_j)}, \forall o \in \mathcal{O} \\ p_j = \prod_{k=1}^{m} \sum_{w_i \in \mathcal{W} \, s.t. \, w_i \cap \mathcal{O}_k = w_j \cap \mathcal{O}_k} p_i \qquad \forall w_j \in \mathcal{W} \end{cases}$$

The first two types of constraints enforce a probability distribution over the set of possible worlds. The third type of constraint ensures that the probability of occurrence o – which is the sum of the probabilities of the worlds containing o – is equal to its relative probability $p^*(o)$ weighted by $\frac{Weight(o)}{\sum_{o_j \in C(o)} Weight(o_j)}$, the latter being the weight of o divided by the sum of the weights of the occurrences conflicting with o. Note that: (i) the value on the right-hand side of the third type of constraint decreases as the amount of conflict increases, (ii) if an occurrence o is not conflicting with any other occurrence, then its probability $\sum_{w_i \in \mathcal{W} \, s.t. \, o \in w_i} p_i$ is equal to $p^*(o)$, i.e. the probability returned by the stochastic automaton. The last kind of constraint reflects the independence between segments. In general $NLC(v, \ell)$ might admit multiple solutions.

Example 4.5: Consider a single-segment video consisting of frames $f_1, \ldots, f_9$ shown in Figure 2. Suppose the three occurrences o_1, o_2, o_3 have been detected with relative probabilities 0.3, 0.6, and 0.5, respectively. Suppose the weights of o_1, o_2, o_3 are 1, 2, 3, respectively. Five worlds are possible in this case: $w_0 = \emptyset$, $w_1 = \{o_1\}$, $w_2 = \{o_2\}$, $w_3 = \{o_3\}$, and $w_4 = \{o_1, o_3\}$. Then, $NLC(v, \ell)$ is:[5]

$$\begin{array}{l} p_i \geq 0 \qquad 0 \leq i \leq 4 \\ p_0 + p_1 + p_2 + p_3 + p_4 = 1 \\ p_1 + p_4 = 0.3 \cdot \frac{1}{3} \\ p_2 = 0.6 \cdot \frac{1}{3} \\ p_3 + p_4 = 0.5 \cdot \frac{3}{5} \end{array}$$

which has multiple solutions. One solution is $p_0 = 0.4$, $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$, $p_4 = 0$. Another solution is $p_0 = 0.5$, $p_1 = 0$, $p_2 = 0.2$, $p_3 = 0.2$, $p_4 = 0.1$.

In the rest of the paper, we assume that $NLC(v, \ell)$ is solvable.[6] We say that a sequence $S = \langle (f_1, s_1), \ldots, (f_n, s_n) \rangle$ *occurs in* a video v w.r.t. a labeling ℓ iff $\langle f_1, \ldots, f_n \rangle$ is a contiguous subsequence of

5. For brevity, we do not explicitly list the independence constraints.

6. This can be easily checked via both a non-linear constraint solver, as well as methods developed in the next section.

v and $s_i \in \ell(f_i)$ for $1 \leq i \leq n$. We give two semantics for S to be unexplained in a world $w \in \mathcal{W}$:

1) S is *totally unexplained* in w, denoted $w \not\models_T S$, iff $\forall (f_i, s_i) \in S, \nexists o \in w, (f_i, s_i) \in o$;
2) S is *partially unexplained* in w, denoted $w \not\models_P S$, iff $\exists (f_i, s_i) \in S, \nexists o \in w, (f_i, s_i) \in o$.

Intuitively, S is totally (resp. partially) unexplained in w iff w does not explain every (resp. at least one) symbol of S. We now define the probability that a sequence occurring in a video is totally or partially unexplained.

Definition 4.4: Let v be a video, ℓ a labeling, and S a sequence occurring in v w.r.t. ℓ. The probability interval that S is totally unexplained in v w.r.t. ℓ is $\mathcal{I}_T(S) = [l, u]$, where:

$$\begin{array}{l} l = \textbf{minimize} \sum_{w_i \in \mathcal{W} \ s.t. \ w_i \not\models_T S} p_i \\ \qquad \textbf{subject to } NLC(v, \ell) \\ u = \textbf{maximize} \sum_{w_i \in \mathcal{W} \ s.t. \ w_i \not\models_T S} p_i \\ \qquad \textbf{subject to } NLC(v, \ell) \end{array}$$

The probability interval that S is partially unexplained in v w.r.t. ℓ is $\mathcal{I}_P(S) = [l', u']$, where l', u' are derived in exactly the same way as l, u above by replacing the $\not\models_T$ symbols in the above optimization problems by $\not\models_P$.

Thus, the probability that a sequence S occurring in v is totally (resp. partially) unexplained w.r.t. to a solution of $NLC(v, \ell)$ is the sum of the probabilities of the worlds in which S is totally (resp. partially) unexplained. As $NLC(v, \ell)$ may have multiple solutions, we find the tightest interval $[l, u]$ (resp. $[l', u']$) containing this probability for any solution. Different criteria can be used to infer a value from an interval $[l, u]$, e.g. the MIN l, the MAX u, the average (i.e., $(l + u)/2$), etc. Clearly, the only requirement is that this value has to be in $[l, u]$. In the rest of the paper we assume that one of the above criteria has been chosen — $\mathcal{P}_T(S)$ (resp. $\mathcal{P}_P(S)$) denotes the probability that S is totally (resp. partially) unexplained.

Proposition 4.1: Consider two sequences S_1 and S_2 occurring in a video. If S_1 is a subsequence of S_2, then $\mathcal{P}_T(S_1) \geq \mathcal{P}_T(S_2)$ and $\mathcal{P}_P(S_1) \leq \mathcal{P}_P(S_2)$.

We now define totally and partially unexplained activity occurrences.

Definition 4.5 (Unexplained activity occurrence): Let v be a video, ℓ a labeling, $\tau \in [0, 1]$ a probability threshold, and $L \in \mathbb{N}^+$ a length threshold. Then,

- a *totally unexplained activity occurrence* is a sequence S occurring in v s.t. (i) $\mathcal{P}_T(S) \geq \tau$, (ii) $|S| \geq L$, and (iii) S is maximal, i.e., there does not exist a sequence $S' \neq S$ occurring in v s.t. S is a subsequence of S', $\mathcal{P}_T(S') \geq \tau$, and $|S'| \geq L$.
- a *partially unexplained activity occurrence* is a sequence S occurring in v s.t. (i) $\mathcal{P}_P(S) \geq \tau$, (ii) $|S| \geq L$, and (iii) S is minimal, i.e., there does not exist a sequence $S' \neq S$ occurring in v s.t. S' is a subsequence of S, $\mathcal{P}_P(S') \geq \tau$, and $|S'| \geq L$.

In the definition above, L is the minimum length a sequence must be for it to be considered a possible unexplained activity occurrence. Totally unexplained activities (TUAs for short) S have to be maximal because once we find S, any sub-sequence of it is (totally) unexplained with probability greater than equal to that of S. On the other hand, partially unexplained activities (PUAs for short) S' have to be minimal because once we find S', any super-sequence of it is (partially) unexplained with probability greater than or equal to that of S'.

Intuitively, an unexplained activity occurrence is a sequence of action symbols that are observed in the video and poorly explained by the known activity models. Such sequences might correspond to unknown variants of known activities or to entirely new – and unknown – activities.

An *Unexplained Activity Problem* (UAP) instance is a 4-tuple $I = \langle v, \ell, \tau, L \rangle$ where v is a video, ℓ is a labeling, $\tau \in [0, 1]$ is a probability threshold, and $L \in \mathbb{N}^+$ is a length threshold. We want to find the sets $\mathcal{A}^{tu}(I)$ and $\mathcal{A}^{pu}(I)$ of all totally and partially unexplained activities, respectively. When I is clear from context, we will drop it.

The following definition introduces the top-k totally and partially unexplained activities. Intuitively, these are k unexplained activities having maximum probability.

Definition 4.6 (Top-k unexplained activities): Consider an UAP instance and let $k \in \mathbb{N}^+$. $\mathcal{A}^{tu}_k \subseteq \mathcal{A}^{tu}$ (resp. $\mathcal{A}^{pu}_k \subseteq \mathcal{A}^{pu}$) is a set of top-$k$ totally (resp. partially) unexplained activities iff $|\mathcal{A}^{tu}_k| = \min\{k, |\mathcal{A}^{tu}|\}$ (resp. $|\mathcal{A}^{pu}_k| = \min\{k, |\mathcal{A}^{pu}|\}$), and $\forall S \in \mathcal{A}^{tu}_k, \forall S' \in \mathcal{A}^{tu} - \mathcal{A}^{tu}_k$ (resp. $\forall S \in \mathcal{A}^{pu}_k, \forall S' \in \mathcal{A}^{pu} - \mathcal{A}^{pu}_k$) $\mathcal{P}_T(S) \geq \mathcal{P}_T(S')$ (resp. $\mathcal{P}_P(S) \geq \mathcal{P}_P(S')$).

Suppose we have an UAP instance. For any $S, S' \in \mathcal{A}^{tu}$ (resp. $S, S' \in \mathcal{A}^{pu}$), we write $S =_T S'$ (resp. $S =_P S'$) iff $\mathcal{P}_T(S) = \mathcal{P}_T(S')$ (resp. $\mathcal{P}_P(S) = \mathcal{P}_P(S')$). Obviously, $=_T$ (resp. $=_P$) is an equivalence relation and determines a set $\mathcal{C}^{tu}$ (resp. $\mathcal{C}^{pu}$) of equivalence classes. For any equivalence class $C \in \mathcal{C}^{tu}$ (resp. $C \in \mathcal{C}^{pu}$) we define $\mathcal{P}_T(C)$ (resp. $\mathcal{P}_P(C)$) as the (unique) probability of the sequences in C.

The top-k totally and partially unexplained classes are the k classes having maximum probability. Compared with the top-k unexplained activities, here we want to return *all* the unexplained activities having the k highest probabilities.

Definition 4.7 (Top-k unexplained classes): Consider an UAP instance and let $k \in \mathbb{N}^+$. $\mathcal{C}^{tu}_k \subseteq \mathcal{C}^{tu}$ (resp. $\mathcal{C}^{pu}_k \subseteq \mathcal{C}^{pu}$) is the set of top-$k$ totally (resp. partially) unexplained classes iff $|\mathcal{C}^{tu}_k| = \min\{k, |\mathcal{C}^{tu}|\}$ (resp. $|\mathcal{C}^{pu}_k| = \min\{k, |\mathcal{C}^{pu}|\}$), and $\forall C \in \mathcal{C}^{tu}_k, \forall C' \in \mathcal{C}^{tu} - \mathcal{C}^{tu}_k$ (resp. $\forall C \in \mathcal{C}^{pu}_k, \forall C' \in \mathcal{C}^{pu} - \mathcal{C}^{pu}_k$) $\mathcal{P}_T(C) > \mathcal{P}_T(C')$ (resp. $\mathcal{P}_P(C) > \mathcal{P}_P(C')$).

5 Properties of UAPs

This section derives properties of UAPs that can be leveraged (in the next section) to devise efficient algorithms to solve UAPs. We first show an interesting property concerning the solution of $NLC(v, \ell)$ (some subsequent results rely on it); then, in the following two subsections, we consider specific properties for totally and partially unexplained activities.

For a given video v and labeling ℓ, we now show that if $\langle(v_1, \ell_1), \dots, (v_m, \ell_m)\rangle$ is a CBP, then we can find the solutions of the *non-linear* constraints $NLC(v, \ell)$ by solving *m smaller sets of linear constraints.*[7] We define $LC(v, \ell)$ as the set of linear constraints of $NLC(v, \ell)$ (thus, we include all the constraints of Definition 4.3 except for the last kind). Henceforth, we use $\mathcal{W}$ to denote $\mathcal{W}(v, l)$ and $\mathcal{W}_i$ to denote $\mathcal{W}(v_i, \ell_i)$, $1 \leq i \leq m$. A solution of $NLC(v, \ell)$ is a mapping $\mathcal{P} : \mathcal{W} \to [0, 1]$ which satisfies $NLC(v, \ell)$. Likewise, a solution of $LC(v_i, \ell_i)$ is a mapping $\mathcal{P}_i : \mathcal{W}_i \to [0, 1]$ which satisfies $LC(v_i, \ell_i)$. It is important to note that $\mathcal{W} = \{w_1 \cup \ldots \cup w_m \mid w_i \in \mathcal{W}_i, 1 \leq i \leq m\}$.

Theorem 1: Let v be a video, ℓ a labeling, and $\langle(v_1, \ell_1), \dots, (v_m, \ell_m)\rangle$ a CBP. $\mathcal{P}$ is a solution of $NLC(v, \ell)$ iff $\forall i \in [1, m]$ there exists a solution $\mathcal{P}_i$ of $LC(v_i, \ell_i)$ s.t. $\mathcal{P}(\bigcup_{i=1}^{m} w_i) = \prod_{i=1}^{m} \mathcal{P}_i(w_i)$ for every $w_1 \in \mathcal{W}_1, \dots, w_m \in \mathcal{W}_m$.

The following example illustrates the previous theorem.

Example 5.1: Consider the video v and the labeling ℓ of Example 4.3 (cf. Figure 2). As shown in Example 4.4, one possible CBP of v and ℓ is $\langle(v_1, \ell_1), (v_2, \ell_2)\rangle$, where $v_1 = \langle f_1, \dots, f_9\rangle$, $v_2 = \langle f_{10}, \dots, f_{16}\rangle$, ℓ_1 and ℓ_2 are the restrictions of ℓ to v_1 and v_2, respectively. Theorem 1 says that for each solution $\mathcal{P}$ of $NLC(v, \ell)$, there is a solution $\mathcal{P}_1$ of $LC(v_1, \ell_1)$ and a solution $\mathcal{P}_2$ of $LC(v_2, \ell_2)$ s.t. $\mathcal{P}(w_1 \cup w_2) = \mathcal{P}_1(w_1) \times \mathcal{P}(w_2)$ for every $w_1 \in \mathcal{W}_1, w_2 \in \mathcal{W}_2$, and vice versa.

Consider a video v and a labeling ℓ, and let $\langle(v_1, \ell_1), \dots, (v_m, \ell_m)\rangle$ be a CBP. Given a sequence $S = \langle(f_1, s_1), \dots, (f_q, s_q)\rangle$ occurring in v, we say that $v_i, v_{i+1}, \dots, v_{i+n}$ ($1 \leq i \leq i + n \leq m$) are the sub-videos containing S iff $f_1 \in v_i$ and $f_q \in v_{i+n}$. In other words, S spans the sub-videos $v_i, v_{i+1}, \dots, v_{i+n}$: it starts at a point in sub-video v_i (as v_i contains the first frame of S) and ends at some point in sub-video v_{i+n} (as v_{i+n} contains the last frame of S). S_k denotes the *projection* of S on the k-th sub-video v_k ($i \leq k \leq i + n$), that is, the subsequence of S containing all the pairs $(f, s) \in S$ with $f \in v_k$.

Example 5.2: Suppose we have a video $v = \langle f_1, \dots, f_{21}\rangle$ and a labeling ℓ such that $\ell(f_i) = \{s_i\}$ for $1 \leq i \leq 21$. In addition, suppose 8 occurrences are detected as shown in Figure 3. Consider the CBP $\langle(v_1, \ell_1), (v_2, \ell_2), (v_3, \ell_3), (v_4, \ell_4)\rangle$, where $v_1 = \{f_1, \dots, f_5\}$, $v_2 = \{f_6, \dots, f_{10}\}$, $v_3 = \{f_{11}, \dots, f_{16}\}$, $v_4 = \{f_{17}, \dots, f_{21}\}$, and ℓ_i is the restrictions of ℓ to v_i, for $1 \leq i \leq 4$.

Consider now the sequence $S = \langle(f_8, s_8), \dots, (f_{14}, s_{14})\rangle$ occurring in v. Then, v_2 and v_3 are the sub-videos containing S. Moreover, S_2 denotes $\langle(f_8, s_8), \dots, (f_{10}, s_{10})\rangle$, and S_3 denotes $\langle(f_{11}, s_{11}), \dots, (f_{14}, s_{14})\rangle$.

7. This therefore yields two benefits: first it allows us to solve a smaller set of constraints, and second, it allows us to solve linear constraints which are usually easier to solve than nonlinear ones.

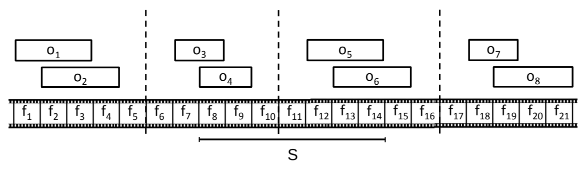

o1
o2
o3
o4
o5
o6
o7
o8
f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 f13 f14 f15 f16 f17 f18 f19 f20 f21
S

Fig. 3: Conflict-Based Partitioning of a video

5.1 Totally unexplained activities

The following theorem says that we can compute $\mathcal{I}_T(S)$ by solving LC (which are linear constraints) for each sub-video containing S (instead of solving a non-linear set of constraints for the whole video).

Theorem 2: Consider a video v and a labeling ℓ. Let $\langle(v_1, \ell_1), \dots, (v_m, \ell_m)\rangle$ be a CBP and $\langle v_i, \dots, v_{i+n}\rangle$ be the sub-videos containing a sequence S occurring in v. For $i \leq k \leq i + n$, let

$$\begin{aligned} l_k = \mathbf{minimize} & \textstyle\sum_{w_h \in \mathcal{W}_k \ s.t.\ w_h \not\models_T S_k} p_h \\ & \mathbf{subject\ to}\ LC(v_k, \ell_k) \\ u_k = \mathbf{maximize} & \textstyle\sum_{w_h \in \mathcal{W}_k \ s.t.\ w_h \not\models_T S_k} p_h \\ & \mathbf{subject\ to}\ LC(v_k, \ell_k) \end{aligned}$$

If $\mathcal{I}_T(S) = [l, u]$, then $l = \prod_{k=i}^{i+n} l_k$ and $u = \prod_{k=i}^{i+n} u_k$.

The following example illustrates the theorem above.

Example 5.3: Consider the setting of Example 5.2, which is depicted in Figure 3. By definition, $\mathcal{I}_T(S)$ can be computed by solving the non-linear program of Definition 4.4 for the whole video v. Alternatively, Theorem 2 says that $\mathcal{I}_T(S)$ can be computed as $\mathcal{I}_T(S) = [l_2 \times l_3, u_2 \times u_3]$, where l_2, u_2, l_3, u_3 are computed as defined in Theorem 2, that is, by solving two smaller linear programs for v_2 and v_3.

The following theorem provides a sufficient condition for a pair (f, s) not to be included in any sequence S occurring in v and having $\mathcal{P}_T(S) \geq \tau$.

Theorem 3: Let $\langle v, \ell, \tau, L\rangle$ be a UAP instance. Given (f, s) s.t. $f \in v$ and $s \in \ell(f)$, let $\varepsilon = \displaystyle\sum_{o \in \mathcal{O}\ s.t.\ (f,s) \in o} p^*(o) \cdot \frac{Weight(o)}{\sum_{o_j \in C(o)} Weight(o_j)}$. If $\varepsilon > 1 - \tau$, then there does not exist a sequence S occurring in v s.t. $(f, s) \in S$ and $\mathcal{P}_T(S) \geq \tau$.

If the above condition holds for a pair (f, s), then we say that (f, s) is *sufficiently explained. Note that to check whether a pair (f, s) is sufficiently explained, we do not need to solve any set of linear or non-linear constraints, since ε is computed by simply summing the (weighted) probabilities of the occurrences containing (f, s).* Thus, this result yields a further efficiency. A frame f is *sufficiently explained* iff (f, s) is sufficiently explained for every $s \in \ell(f)$. If (f, s) is sufficiently explained, then it can be disregarded for the purpose of identifying unexplained activity occurrences, and, in addition, this may allow us to disregard entire parts of videos as shown in the example below.

Example 5.4: Consider a UAP instance $\langle v, \ell, \tau, L\rangle$ where $v = \langle f_1, \ldots, f_9\rangle$ and ℓ is s.t. $\ell(f_i) = \{s_i\}$ for $1 \le i \le 9$, as depicted in Figure 4.

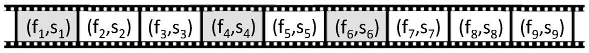


Fig. 4: Sufficiently explained frames in a video.

Suppose $L = 3$ and (f_1, s_1), (f_4, s_4), (f_6, s_6) are sufficiently explained. Even though we could apply the theorem to only a few (f_i, s_i) pairs, we can conclude that no unexplained activity occurrence can be found before f_7, because $L = 3$.

Given a UAP instance $I = \langle v, \ell, \tau, L\rangle$ and a subsequence v' of v, v' is *relevant* iff (i) v' is a contiguous subsequence of v (ii) $|v'| \ge L$, (iii) $\forall f \in v'$, f is not sufficiently explained, and (iv) v' is maximal (i.e., there does not exist $v'' \neq v'$ s.t. v' is a subsequence of v'' and v'' satisfies (i), (ii), (iii)). We use $relevant(I)$ to denote the set of relevant sub-videos.

Theorem 3 entails that relevant sub-videos can be individually considered when looking for totally unexplained activities because there is no totally unexplained activity spanning two different relevant sub-videos.

5.2 Partially unexplained activities

The following theorem states that we can compute $\mathcal{I}_P(S)$ by solving NLC for the sub-video consisting of the segments containing S (instead of solving NLC for the whole video).

Theorem 4: Consider a video v and a labeling ℓ. Let $\langle(v_1, \ell_1), \ldots, (v_m, \ell_m)\rangle$ be a CBP and $\langle v_i, \ldots, v_{i+n}\rangle$ be the sub-videos containing a sequence S occurring in v. Let $v^* = v_i \cdot \ldots \cdot v_{i+n}$ and ℓ^* be a labeling for v^* s.t., for every $f \in v^*$, $\ell^*(f) = \ell(f)$. $\mathcal{I}_P(S)$ computed w.r.t. v and ℓ is equal to $\mathcal{I}_P(S)$ computed w.r.t. v^* and ℓ^*.

We now illustrate the use of the preceding theorem.

Example 5.5: Consider the setting of Example 5.2, which is depicted in Figure 3. By definition, $\mathcal{I}_P(S)$ can be computed by solving the non-linear program of Definition 4.4 for the whole video v. Alternatively, Theorem 4 says that $\mathcal{I}_P(S)$ can be computed by solving the non-linear program of Definition 4.4 for the sub-video $v^* = v_2 \cdot v_3$.

6 TOP-k ALGORITHMS

We now present algorithms to find top-k totally and partially unexplained activities and classes. For ease of presentation, we assume $|\ell(f)| = 1$ for every frame f in a video (this makes the algorithms much more concise – generalization to the case of multiple action symbols per frame is straightforward[8]). Given a video $v = \langle f_1, \ldots, f_n\rangle$, we use $v(i, j)$ ($1 \le i \le j \le n$) to denote the sequence $S = \langle(f_i, s_i), \ldots, (f_j, s_j)\rangle$, where s_k is the only element in $\ell(f_k)$, $i \le k \le j$.

8. Indeed, it suffices to consider the different sequences given by the different action symbols.

6.1 Top-k TUA and TUC

The Top-k TUA algorithm computes a set of top-k totally unexplained activities in a video. Note that:

- at every time, $lowest$ is defined as follows:
$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_T(S) \mid S \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$
- On line 30, "Add S to $TopSol$" works as follows:
  - If $|TopSol| < k$, then S is added to $TopSol$;
  - otherwise, a sequence S' in $TopSol$ having minimum $\mathcal{P}_T(S')$ is replaced by S.

Leveraging Theorem 3, Top-k TUA considers only relevant sub-videos of v individually (line 2). When it finds a sequence $v'(start, end)$ of length at least L having a probability of being totally unexplained greater than $lowest$ (line 5), it makes the sequence maximal by adding frames on the right (lines 7–14). Instead of adding one frame at a time, $v'(start, end)$ is extended by L frames at a time until its probability drops below τ (lines 9–10); a binary search is then performed to find the exact maximum length of the unexplained activity (lines 15–25). Note that, when making the sequence maximal, if at some point the algorithm realizes that the unexplained activity will not have a probability greater than $lowest$ (i.e. the sequence is not a top-k TUA), then the sequence is disregarded and the above process of making the sequence maximal is aborted (lines 12–14 and 19–21). This kind of pruning allows the algorithm to move forward in the video avoiding computing the exact ending frame of the TUA thereby saving time. Throughout the algorithm, $\mathcal{P}_T$ is computed by applying Theorem 2.

Theorem 5: Algorithm Top-k TUA returns a set of top-k totally unexplained activities of the input instance.

Algorithm Top-k TUC modifies Top-k TUA as follows to compute the top-k totally unexplained classes:

- At every time, $lowest$ is defined as follows:
$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_T(C) \mid C \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$
- "Add S to $TopSol$" (line 30) works as follows:
  - If there exists $C \in TopSol$ s.t. $\mathcal{P}_T(C) = \mathcal{P}_T(S)$, then S is added to C;
  - else if $|TopSol| < k$, then the class $\{S\}$ is added to $TopSol$;
  - otherwise the class C in $TopSol$ having minimum $\mathcal{P}_T(C)$ is replaced with $\{S\}$.
- On line 5, $\mathcal{P}_T(v'(start, end)) > lowest$ is replaced with $\mathcal{P}_T(v'(start, end)) \ge lowest$;
- On line 12, $\mathcal{P}_T(v'(start, end)) \le lowest$ is replaced with $\mathcal{P}_T(v'(start, end)) < lowest$;
- On line 19, $\mathcal{P}_T(v'(start, mid)) \le lowest$ is replaced with $\mathcal{P}_T(v'(start, mid)) < lowest$;

The algorithm obtained by applying the modifications above is named Top-k TUC.

Theorem 6: Algorithm Top-k TUC returns the top-k totally unexplained classes of the input instance.

Algorithm 1 Top-k TUA

```
Input: UAP instance $I = \langle v, \ell, \tau, L\rangle$, $k \geq 1$
Output: Top-$k$ totally unexplained activities
 1: $TopSol = \emptyset$
 2: for all $v' \in relevant(I)$ do
 3:     $start = 1$;  $end = L$
 4:     repeat
 5:         if $\mathcal{P}_T(v'(start, end)) \geq \tau \wedge \mathcal{P}_T(v'(start, end)) > lowest$ then
 6:             $end' = end$
 7:             while $end < |v'|$ do
 8:                 $end = \min\{end + L, |v'|\}$
 9:                 if $\mathcal{P}_T(v'(start, end)) < \tau$ then
10:                     break
11:                 else
12:                     if $\mathcal{P}_T(v'(start, end)) \leq lowest$ then
13:                         $end = end + 1$
14:                         go to line 33
15:             $s = \max\{end - L, end'\}$;  $e = end$
16:             while $e \neq s$ do
17:                 $mid = \lceil (s + e)/2 \rceil$
18:                 if $\mathcal{P}_T(v'(start, mid)) \geq \tau$ then
19:                     if $\mathcal{P}_T(v'(start, mid)) \leq lowest$ then
20:                         $end = mid + 1$
21:                         go to line 33
22:                     else
23:                         $s = mid$
24:                 else
25:                     $e = mid - 1$
26:             if $start > 1 \wedge \mathcal{P}_T(v'(start - 1, s)) \geq \tau$ then
27:                 $end = s + 1$
28:                 go to line 33
29:             else
30:                 $S = v'(start, s)$;  Add $S$ to $TopSol$
31:                 $start = start + 1$;  $end = s + 1$
32:         else
33:             $start = start + 1$;   $end = \max\{end, start + L - 1\}$
34:     until $end > |v'|$
35: return $TopSol$
```

6.2 Top-k PUA and PUC

The Top-k PUA algorithm below computes a set of top-k partially unexplained activities in a video. Note that:

- at each time, $lowest$ is defined as follows:

$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_P(S) \mid S \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$

- On line 43, "Add S to $TopSol$" works as follows:
  - If $|TopSol| < k$, then S is added to $TopSol$;
  - otherwise, a sequence in $TopSol$ having minimum $\mathcal{P}_P$ is replaced by S.

To find an unexplained activity, Algorithm Top-k PUA starts with a sequence of length at least L and adds frames to its right until its probability of being partially unexplained is above the threshold. As in the case of Top-k TUA, this is done by adding L frames at a time (lines 5–8) and then performing a binary search (lines 9–27). When performing the binary search, if at some point the algorithm realizes that the partially unexplained activity will not have a probability greater than $lowest$, then the sequence is disregarded and the binary search is aborted (lines 17–19 and lines 24–25). Otherwise, the sequence is shortened on the left making it minimal (lines 28–38) by performing a binary search instead of proceeding one frame at a time. Once again, if the algorithm realizes that the partially unexplained activity will not have a probability greater than $lowest$, then the sequence is disregarded and the shortening process is aborted (lines 34–36). This allows the algorithm to avoid computing the exact starting frame of the PUA, thus saving time. Note that $\mathcal{P}_P$ is computed by applying Theorem 4.

Algorithm 2 Top-k PUA

```
Input: UAP instance $I = \langle v, \ell, \tau, L\rangle$, $k \geq 1$
Output: Top-$k$ partially unexplained activities
 1: $TopSol = \emptyset$;   $start = 1$;   $end = L$
 2: while $end \leq |v|$ do
 3:     if $\mathcal{P}_P(v(start, end)) < \tau$ then
 4:         $end' = end$
 5:         while $end < |v|$ do
 6:             $end = \min\{end + L, |v|\}$
 7:             if $\mathcal{P}_P(v(start, end)) \geq \tau$ then
 8:                 break
 9:         if $\mathcal{P}_P(v(start, end)) \geq \tau$ then
10:             if $\mathcal{P}_P(v(start, end)) > lowest$ then
11:                 $s = \max\{end' + 1, end - L + 1\}$;   $e = end$
12:                 while $e \neq s$ do
13:                     $mid = \lfloor (s + e)/2 \rfloor$
14:                     if $\mathcal{P}_P(v(start, mid)) < \tau$ then
15:                         $s = mid + 1$
16:                     else
17:                         if $\mathcal{P}_P(v(start, mid)) \leq lowest$ then
18:                             $start = start + 1$;   $end = mid + 1$
19:                             go to line 2
20:                         else
21:                             $e = mid$
22:                 $end = e$
23:             else
24:                 $start = start + 1$;   $end = end + 1$
25:                 go to line 2
26:         else
27:             return  $TopSol$
28:     $s' = start$;  $e' = end - L + 1$
29:     while $e' \neq s'$ do
30:         $mid = \lceil (s' + e')/2 \rceil$
31:         if $\mathcal{P}_P(v(mid, end)) < \tau$ then
32:             $e' = mid - 1$
33:         else
34:             if $\mathcal{P}_P(v(mid, end)) \leq lowest$ then
35:                 $start = mid + 1$;   $end = end + 1$
36:                 go to line 2
37:             else
38:                 $s' = mid$
39:     if $\mathcal{P}_P(v(s', end - 1)) \geq \tau \wedge |v(s', end - 1)| \geq L$ then
40:         $start = s' + 1$;   $end = end + 1$
41:         go to line 2
42:     else
43:         $S = v(s', end)$;  Add $S$ to $TopSol$
44:         $start = s' + 1$;   $end = end + 1$
45: return  $TopSol$
```

Theorem 7: Algorithm Top-k PUA returns a set of top-k partially unexplained activities of the input instance.

Algorithm Top-k PUC modifies Top-k PUA as follows to compute the top-k partially unexplained classes:

- At every time, $lowest$ is defined as follows:

$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_P(C) \mid C \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$

- "Add S to $TopSol$" (line 43) works as follows:
  - If there exists $C \in TopSol$ s.t. $\mathcal{P}_P(C) = \mathcal{P}_P(S)$, then S is added to C;
  - else if $|TopSol| < k$, then the class $\{S\}$ is added to $TopSol$;
  - otherwise the class C in $TopSol$ having minimum $\mathcal{P}_P(C)$ is replaced with $\{S\}$.
- On line 10, $\mathcal{P}_P(v(start, end)) > lowest$ is replaced with $\mathcal{P}_P(v(start, end)) \geq lowest$;
- On line 17, $\mathcal{P}_P(v(start, mid)) \leq lowest$ is replaced with $\mathcal{P}_P(v(start, mid)) < lowest$;
- On line 34, $\mathcal{P}_P(v(mid, end)) \leq lowest$ is replaced with $\mathcal{P}_P(v(mid, end)) < lowest$;

The algorithm obtained by applying the modifications above is named Top-k PUC.

Theorem 8: Algorithm Top-k PUC returns the top-k partially unexplained classes of the input instance.

7 Experimental Evaluation

Our prototype implementation of the proposed framework consists of (i) an *image processing library*, which performs low-level processing of video frames, including object tracking and classification; (ii) a *video labeler*, which maps frames to action symbols based on the output of the image processing stage (i.e. gives a labeling of the video), (iii) an *activity recognition algorithm* based on [1] which identifies all possible occurrences of known activities (specified by a set $\mathcal{A}$) in the input video, (iv) a *UAP engine*, which implements Algorithms Top-k TUA, Top-k PUA, Top-k TUC and Top-k PUC in 10,000 lines of Java code.

We experimentally evaluated our framework in terms of both running time and accuracy on two datasets: (i) a video we shot by monitoring a university parking lot, and (ii) a benchmark dataset about video surveillance in an airport [23].

7.1 Parking lot surveillance video

The set $\mathcal{A}$ defined in this case includes activities such as parking a car, people passing, and other "known" activities we expect to occur in a parking lot.

We compared Algorithms Top-k TUA and Top-k PUA against "naïve" algorithms which are the same as Top-k TUA and Top-k PUA but do not exploit the optimizations provided by the theorems in Section 5.

Figures 5 and 6 show that Top-k TUA and Top-k PUA significantly outperform the naïve algorithms which are not able to scale beyond videos of length 15 and 10 minutes for totally and partially unexplained activities, respectively (with longer videos, the naïve algorithms did not terminate in 3 hours). Figures 7a and 8a zoom in on the running times for Algorithms Top-k TUA and Top-k PUA, respectively. The runtimes in Figure 5 when $k = 5$ and $k = All$ are almost the same (the two curves are indistinguishable) because, up to 15 minutes, there were at most 5 totally unexplained activities in the video. A similar argument applies to Figure 6.

We also evaluated how the different parameters that define an UAP instance affect the running time by varying the values of each parameter while keeping the others fixed to a default value.

Runtime of Top-k TUA. Table 1 reports the values we considered for each parameter along with the corresponding default value.

Parameter	Values	Default value
k	1, 2, 5, All	All
τ	0.4, 0.6, 0.8	0.6
L	160, 200, 240, 280	200
# worlds	7 E+04, 4 E+05, 2 E+07	2 E+07

TABLE 1: Parameter values used in the experiments for Algorithm Top-k TUA (parking lot dataset).

For example, Table 1 says that we measured the running times to find the top-1, top-2, top-5, and all totally unexplained activities (as the video length increases) while keeping $\tau = 0.6$, $L = 200$, $\#worlds = 2E + 07$.

Varying k. Figure 7a shows that lower values of k give lower running times. As discussed in the preceding section, Algorithm Top-k TUA can can infer that some sequences are not going to be top-k TUAs and quickly prune: this is effective with lower values of k because the probability threshold to enter the current Top-k TUAs (i.e., $lowest$ in Algorithm Top-k TUA) is higher, thus it gets more restrictive to be added to the current Top-k TUAs and the pruning applied by Algorithm Top-k TUA becomes more effective.

Varying τ. Figure 7b shows that the runtime decreases as the probability threshold grows. Intuitively, this is because higher probability thresholds are a stricter requirement for a sequence to be totally unexplained, so Algorithm Top-k TUA can prune more.

Varying L. Figure 7c shows that higher values of L yield lower running times, though there is not a big difference

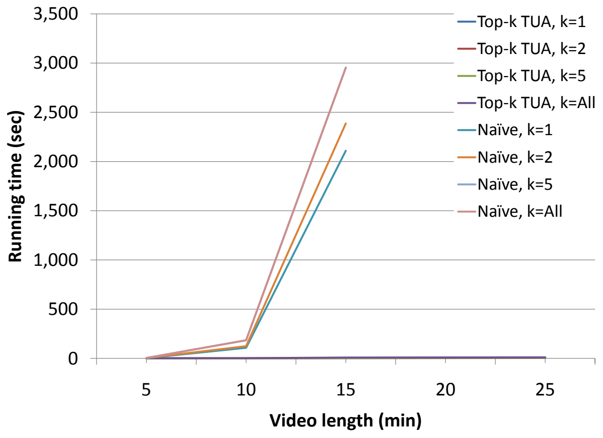


Fig. 5: Algorithm Top-k TUA vs. Naïve.

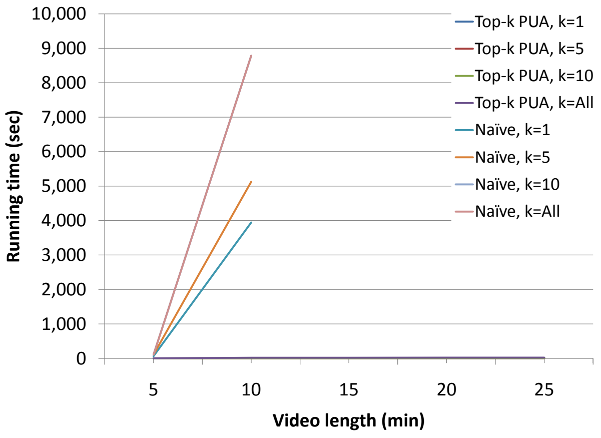
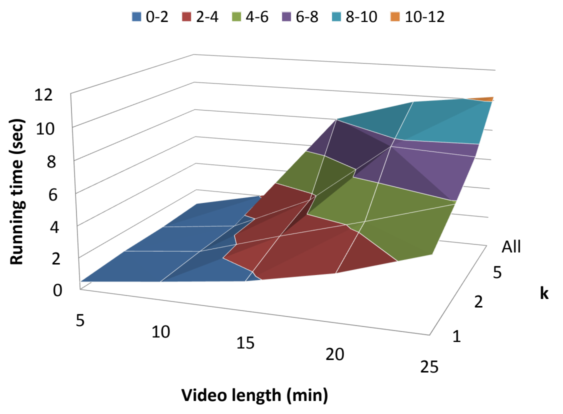
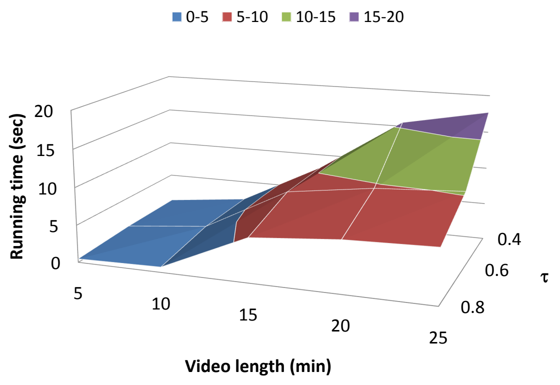


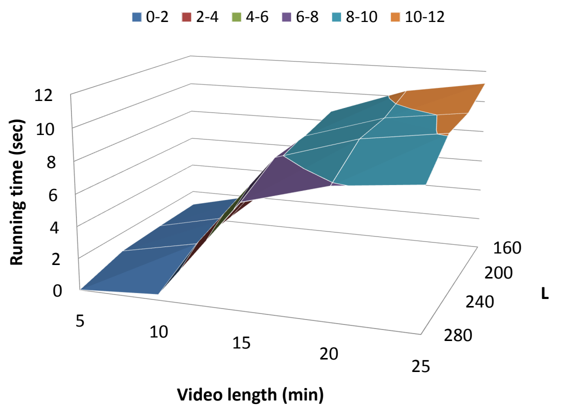
Fig. 6: Algorithm Top-k PUA vs. Naïve.



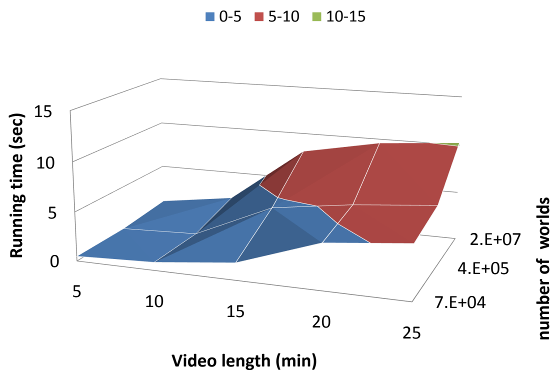
(a) Varying k ($\tau = 0.6, L = 200$)



(b) Varying τ ($k = All, L = 200$)



(c) Varying L ($\tau = 0.6, k = All$)



(d) Varying the number of worlds ($\tau = 0.6, k = All, L = 200$)

Fig. 7: Running time of Algorithm Top-k TUA on the parking lot dataset.

between $L = 200$ and $L = 240$.

Varying Number of Possible Worlds. Finally, Figure 7d shows that more possible worlds leads to higher running times. However, note that big differences in the number of possible worlds yield small differences in running times, hence Algorithm Top-k TUA is able to scale well (this is due to the application of Theorem 2 to compute $\mathcal{P}_T(S)$).

Runtime of Top-k PUA. When analyzing Top-k PUA, we used the same parameter values in Table 1 except for k whose values were $1, 5, 10, All$ with default value All.

Varying k. The runtimes for $k = 1, 5, 10$ differ slightly from each other and are much lower than when all PUAs had to be found (Figure 8a).

Varying τ. Figure 8b shows that the runtimes do not change much for different values of τ.

Varying L. Figure 8c shows that higher values of L lead to lower runtimes.

Varying Number of Possible Worlds. Figure 8d shows that higher numbers of possible worlds lead to higher runtimes. As with TUAs, the runtime of Algorithm Top-k PUA increases reasonably despite the steep growth of possible worlds. Moreover, runtimes of Top-k PUA are higher than for Top-k TUA because computing $\mathcal{P}_P(S)$ requires solving a non-linear program whereas $\mathcal{P}_T(S)$ requires solving linear programs.

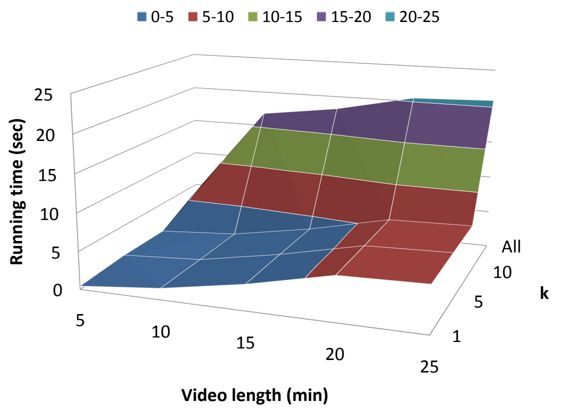
Precision/Recall. In order to assess accuracy, we compared the output of Algorithms Top-k TUA and Top-k PUA against ground truth provided by 8 human annotators who were taught the meaning of graphical representations of activities in $\mathcal{A}$ (e.g. Figure 1). They were asked to identify the totally and partially unexplained sequences in the video w.r.t. $\mathcal{A}$. We ran Top-k TUA and Top-k PUA with values of the probability threshold τ ranging from 0.4 to 0.8, looking for *all* totally and partially unexplained activities in the video (L was set to 200). We use $\{S_i^a\}_{i\in[1,m]}$ to denote the set of unexplained sequences returned by our algorithms and $\{S_j^h\}_{j\in[1,n]}$ to denote the set of sequences flagged as unexplained by human annotators. Precision and recall were computed as:

$$P = \frac{|\{S_i^a | \exists S_j^h \ s.t. \ S_i^a \approx S_j^h\}|}{m}$$

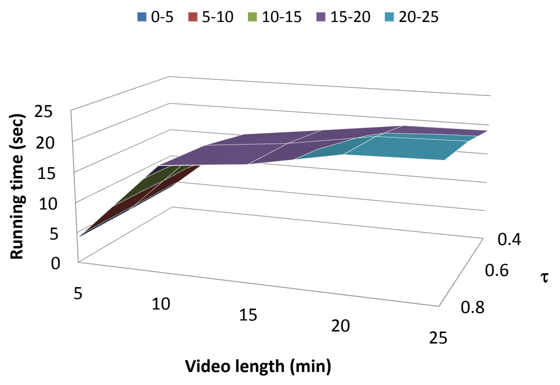
$$R = \frac{|\{S_j^h | \exists S_i^a \ s.t. \ S_i^a \approx S_j^h\}|}{n}$$

where $S_i^a \approx_p S_j^h$ means that S_i^a and S_j^h overlap by a percentage no smaller than 75%.

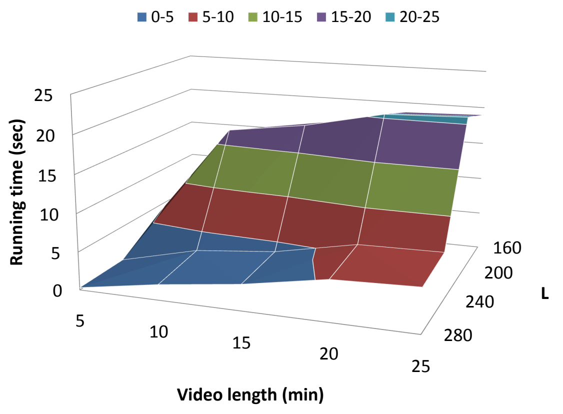
Figure 9 shows the precision/recall graph.



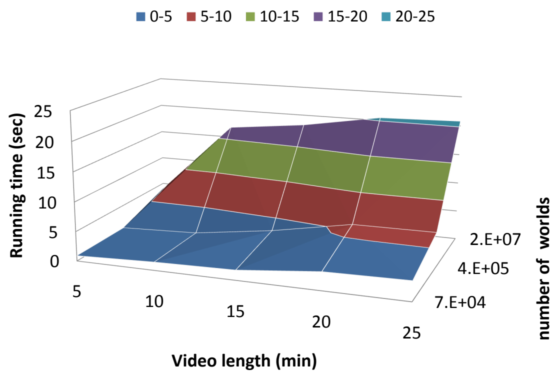
(a) Varying k ($\tau = 0.6, L = 200$)



(b) Varying τ ($k = All, L = 200$)



(c) Varying L ($\tau = 0.6, k = All$)



(d) Varying the number of worlds ($\tau = 0.6, k = All, L = 200$)

Fig. 8: Running time of Algorithm Top-k PUA on the parking lot dataset.

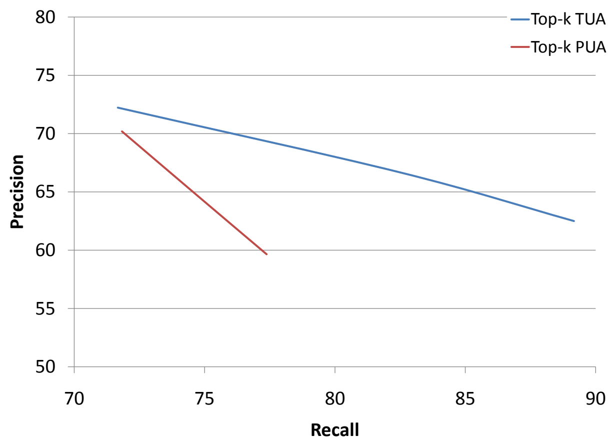


Fig. 9: Precision and Recall for Algorithms Top-k TUA and Top-k PUA (parking lot dataset).

τ	Precision	Recall
0.4	62.5	89.17
0.6	66.67	82.5
0.8	72.22	71.67

TABLE 2: Precision and recall of Algorithm Top-k TUA on the parking lot dataset.

τ	Precision	Recall
0.4	59.65	77.38
0.6	64.91	74.6
0.8	70.18	71.83

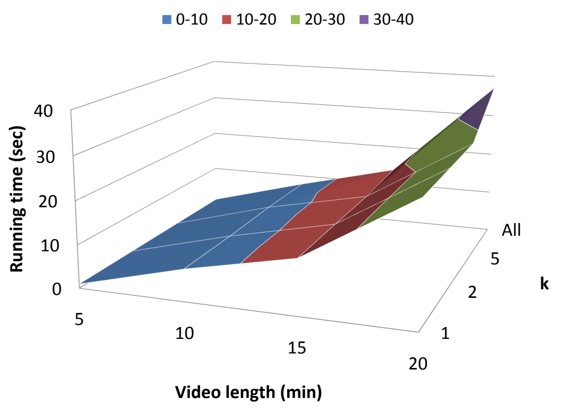
TABLE 3: Precision and recall of Algorithm Top-k PUA on the parking lot dataset.

Precision and recall when $\tau = 0.4, 0.6, 0.8$ are shown in Tables 2 and 3, and show that the framework achieved a good accuracy.

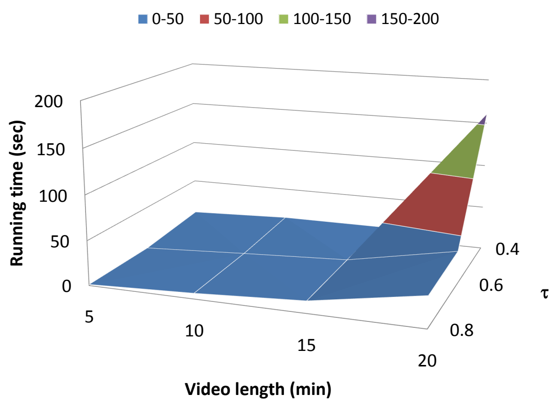
7.2 Airport surveillance video

We also tested our algorithms with an airport video surveillance dataset [23].

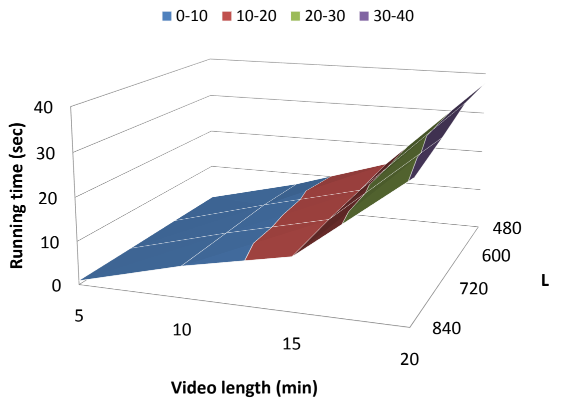
Runtime of Top-k TUA. This data set is far more complex (in terms of number of possible worlds) than the parking lot data set - the "naïve" algorithms did not terminate in a reasonable amount of time, even with a video of 5 minutes. We therefore do not show the running times of the naïve algorithms. As in the case of the parking lot data set, we



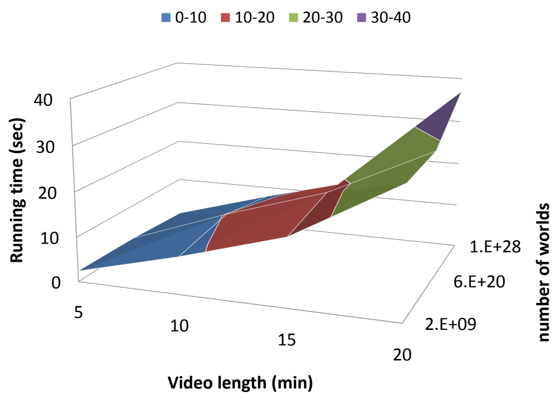
(a) Varying k ($\tau = 0.6, L = 600$)



(b) Varying τ ($k = \mathit{All}, L = 600$)



(c) Varying L ($\tau = 0.6, k = \mathit{All}$)



(d) Varying the number of worlds ($\tau = 0.6, k = \mathit{All}, L = 200$)

Fig. 10: Running time of Algorithm Top-k TUA on the airport dataset.

varied the k, τ, L, $\#\ worlds$ parameters, as shown in Table 4.

Parameter	Values	Default value
k	1, 2, 5, All	All
τ	0.4, 0.6, 0.8	0.6
L	480, 600, 720, 840	600
# worlds	2 E+09, 6 E+20, 1 E+28	1 E+28

TABLE 4: Parameter values used in the experiments for Algorithm Top-k TUA (airport dataset).

Varying k. Figure 10a shows that Top-k TUA's runtime varies little with k when the video is up to 15 minutes long. After that, the runtime for $k = 1, 2, 5$ are comparable, but the runtime for $k = \mathit{All}$ starts to diverge from them.

Varying τ. Figure 10b shows that the runtime when $\tau = 0.4$ is much higher than when $\tau = 0.6$ and $\tau = 0.8$ (the latter two cases do not show substantial differences in running time).

Varying L. Figure 10c shows that higher values of L yield lower runtimes. Though the difference is small for videos under 15 minutes, it becomes marked for 20 minute long videos.

Varying Number of Possible Worlds. Figure 10d shows that runtimes for different numbers of possible worlds are initially close (up to 15 minutes); then, the runtime for 1 E+28 possible worlds gets higher. There is only a moderate increase in runtime corresponding to a huge increase of the number of possible worlds — hence, Top-k TUA is able to scale well when the video gets substantially more complex.

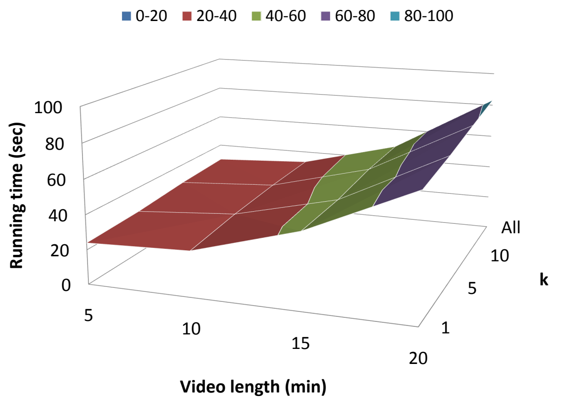
Runtime of Top-k PUA. We conducted experiments with $k = 1, 5, 10, \mathit{All}$ - other parameters were varied according to Table 4.

Varying k. Figure 11a shows that the runtime decreases as k decreases.

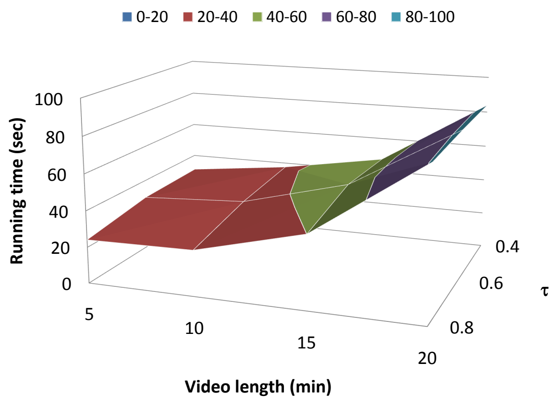
Varying τ. Figure 11b shows that the runtimes for $\tau = 0.4$ and $\tau = 0.6$ are similar and higher than the runtime for $\tau = 0.8$.

Varying L. Figure 11c shows that lower values of L give higher running times. The runtimes are similar for $L = 480$ and $L = 600$ (the number of PUAs found in the video are similar in both cases). Execution times are lower for $L = 720$ and much lower for $L = 800$ (in this case, the number of PUAs found in the video is approximately half the number of PUAs found with $L = 480$ and $L = 600$).

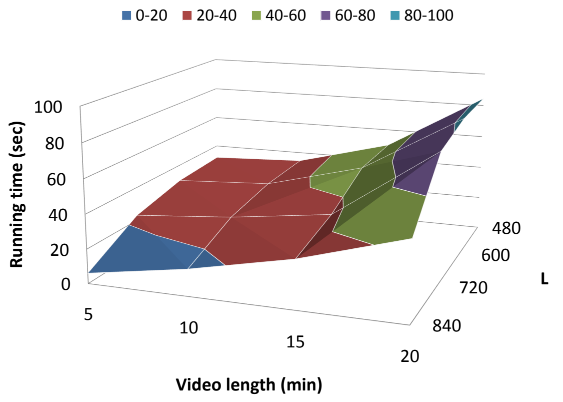
Varying Number of Possible Worlds. Figure 11d shows that though the runtime grows with the number of possible worlds, Top-k PUA responds well to the steep growth of the number of possible worlds.



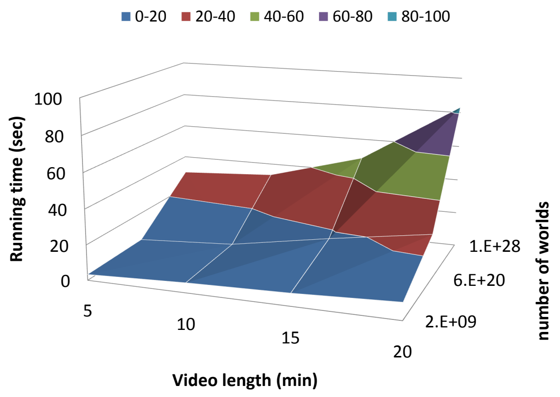
(a) Varying k ($\tau = 0.6, L = 600$)



(b) Varying τ ($k = \mathit{All}, L = 600$)



(c) Varying L ($\tau = 0.6, k = \mathit{All}$)



(d) Varying the number of worlds ($\tau = 0.6, k = \mathit{All}, L = 200$)

Fig. 11: Running time of Algorithm Top-k PUA on the airport dataset.

Precision/Recall. We evaluated the accuracy of Top-k TUA (resp. Top-k PUA) in the same way as for the parking lot data set. The precision/recall graph is reported in Figure 12 and shows that we achieved high accuracy (see also Tables 5 and 6).

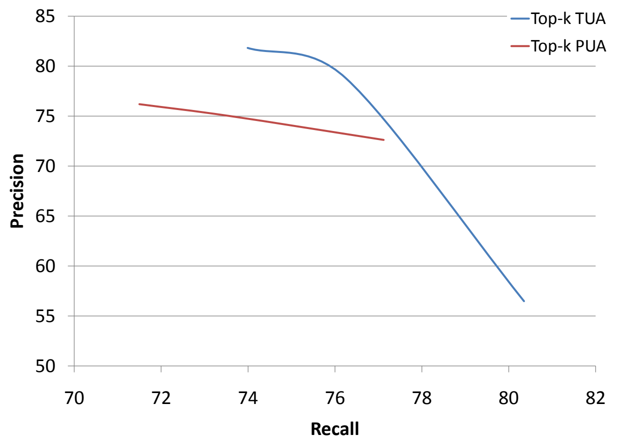


Fig. 12: Precision and Recall for Algorithms Top-k TUA and Top-k PUA (airport dataset).

τ	Precision	Recall
0.4	56.48	80.35
0.6	78.79	76.25
0.8	81.82	73.99

TABLE 5: Precision and recall of Algorithm Top-k TUA on the airport dataset.

τ	Precision	Recall
0.4	72.62	77.12
0.6	75	73.59
0.8	76.19	71.5

TABLE 6: Precision and recall of Algorithm Top-k PUA on the airport dataset.

7.3 Experimental Conclusions

Our experiments show that:
(i) *Runtime increases with video length* (because there are more possible worlds, causing $LC(v, \ell)$ and $NLC(v, \ell)$ to have more variables and constraints). Despite the enormous blow-up in the number of possible worlds, our algorithms perform very well.
(ii) *Runtime increases with the number of totally or partially unexplained activities present in the video.* This is because determining the exact endpoints of each TUA (resp. PUA)

is costly. Specifically, determining the exact end frame of a TUA requires computing $\mathcal{P}_T$ many times: when a TUA is found, Top-k TUA (and also Top-k TUC) need to go through the **while** loop of lines 7–14, the binary search in the **while** loop of lines 16–25, and the **if** block of lines 26–31. All these code blocks require $\mathcal{P}_T$ to be computed. Likewise, determining the exact start and end frames of a PUA requires $\mathcal{P}_P$ to be computed many times as Algorithm Top-k PUA (as well as Algorithm Top-k PUC) goes through different loops and binary searches (one to determine the start frame, another to determine the end frame) requiring multiple computations of $\mathcal{P}_P$.

(iii) *In general, the number of TUAs and PUAs in the video decreases as τ and L increase*, because higher values of τ and L are stricter conditions for a sequence to be totally or partially unexplained.

(iv) *Runtime decreases as k decreases* because our algorithms use k intelligently to infer that certain sequences are not going to be in the result (aborting the loops and binary searches mentioned above).

(v) *Precision increases whereas recall decreases as τ increases*. The experimental results have shown that a good compromise can be achieved by setting τ at least 0.6 and that our framework had a good accuracy with both the datasets we considered.

8 Conclusions

Suppose to have a video v and a set $\mathcal{A}$ of "known" activities (normal or suspicious). In this paper, we address the problem of finding subsequences of v that are not "sufficiently well" explained by the activities in $\mathcal{A}$. We formally define what it means for a video sequence to be unexplained by providing the notions of *totally* and *partially* unexplained activities. We propose a possible worlds framework and identify interesting properties that can be leveraged to make the search for unexplained activities highly efficient via intelligent pruning. We leverage these properties to develop the Top-k TUA, Top-k PUA, Top-k TUC, Top-k PUC algorithms to find totally and partially unexplained activities with highest probabilities. We conducted a detailed experimental evaluation over two datasets showing that our approach works well in practice in terms of both running time and accuracy.

References

[1] M. Albanese, V. Moscato, A. Picariello, V. S. Subrahmanian, and O. Udrea, "Detecting stochastically scheduled activities in video," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2007, pp. 1802–1807.

[2] S. Hongeng and R. Nevatia, "Multi-agent event recognition," in *International Conference On Computer Vision (ICCV)*, 2001, pp. 84–93.

[3] N. Vaswani, A. K. R. Chowdhury, and R. Chellappa, ""Shape Activity": A continuous-state hmm for moving/deforming shapes with application to abnormal activity detection," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1603–1616, 2005.

[4] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 1997, pp. 994–999.

[5] N. Oliver, E. Horvitz, and A. Garg, "Layered representations for human activity recognition," in *International Conference on Multimodal Interfaces (ICMI)*, 2002, pp. 3–8.

[6] R. Hamid, Y. Huang, and I. Essa, "Argmode - activity recognition using graphical models," in *Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2003, pp. 38–43.

[7] N. P. Cuntoor, B. Yegnanarayana, and R. Chellappa, "Activity modeling using event probability sequences," *IEEE Transactions on Image Processing*, vol. 17, no. 4, pp. 594–607, 2008.

[8] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan, "Semi-supervised adapted hmms for unusual event detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 611–618.

[9] M. T. Chan, A. Hoogs, J. Schmiederer, and M. Petersen, "Detecting rare events in video using semantic primitives with hmm," in *International Conference on Pattern Recognition (ICPR)*, 2004, pp. 150–154.

[10] T. Xiang and S. Gong, "Video behavior profiling for anomaly detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 893–908, 2008.

[11] J. Kim and K. Grauman, "Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[12] J. Yin, Q. Yang, and J. J. Pan, "Sensor-based abnormal human-activity detection," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1082–1090, 2008.

[13] D. H. Hu, X.-X. Zhang, J. Yin, V. W. Zheng, and Q. Yang, "Abnormal activity recognition based on hdp-hmm models," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2009, pp. 1715–1720.

[14] X.-X. Zhang, H. Liu, Y. Gao, and D. H. Hu, "Detecting abnormal events via hierarchical dirichlet processes," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2009, pp. 278–289.

[15] D. Mahajan, N. Kwatra, S. Jain, P. Kalra, and S. Banerjee, "A framework for activity recognition and detection of unusual activities," in *Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP)*, 2004.

[16] F. Jiang, Y. Wu, and A. K. Katsaggelos, "Detecting contextual anomalies of crowd motion in surveillance video," in *International Conference on Image Processing (ICIP)*, 2009, pp. 1117–1120.

[17] H. Zhong, J. Shi, and M. Visontai, "Detecting unusual activity in video," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 819–826.

[18] C. E. Au, S. Skaff, and J. J. Clark, "Anomaly detection for video surveillance applications," in *International Conference on Pattern Recognition (ICPR)*, 2006, pp. 888–891.

[19] Y. Zhou, S. Yan, and T. S. Huang, "Detecting anomaly in videos from trajectory similarity analysis," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2007, pp. 1087–1090.

[20] A. Mecocci and M. Pannozzo, "A completely autonomous system that learns anomalous movements in advanced videosurveillance applications," in *International Conference on Image Processing (ICIP)*, 2005, pp. 586–589.

[21] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 555–560, 2008.

[22] F. Jiang, J. Yuan, S. A. Tsaftaris, and A. K. Katsaggelos, "Video anomaly detection in spatiotemporal context," in *International Conference on Image Processing (ICIP)*, 2010, pp. 705–708.

[23] "Ninth IEEE international workshop on performance evaluation of tracking and surveillance (PETS 2006) benchmark data," http://www.cvg.rdg.ac.uk/PETS2006/data.html, 2006.